Appendix A

Common Utilities

Microsoft .NET Framework provides many tools to help developers make the best use of the Framework. In the following sections, we document the commonly used subset of .NET tools that we've used throughout this book:

- Assembly Generation Utility (al.exe)
- Assembly Registration Utility (gac.exe)
- MSIL Assembler (ilasm.exe)
- MSIL Disassembler (ildasm.exe)
- C++ Compiler (cl.exe)
- C# Compiler (csc.exe)
- VB Compiler (vbc.exe)
- PE File viewer (dumpbin.exe)
- Type Library Exporter (tlbexp.exe_)
- Type Library Importer (tlbimp.exe)
- XML Schema Definition Tool (xsd.exe)
- Shared Name Utility (sn.exe)
- Web Service Utility (usdl.exe)

Assembly Generation Utility (al.exe)

While al.exe is generally used to generate assemblies with manifest, it can also be used to install assemblies into the Global Assembly Cache (GAC). The following table shows some of the common usages of the Assembly Generation Utility:

/flags: *flags*	Specifies a value for the Flags field in the assembly. • 0x0000: side-by-side compatible. • 0x0010: cannot execute with other versions in the same application domain. • 0x0020: cannot execute with other versions in the same

	process. • 0x0030: cannot execute with other versions on the same computer.
/help or /?	Use to get help for this command.
/install: *filename* or /I: *filename*	Use to install a strongly named assembly into the GAC. See also **/keyfile** and /**keyname.** If /out is specified, do not specify *filename* here.
/keyfile: *keyfilename* or /keyf: *keyfilename*	Use to create shared components. Keyfilename contains a key pair generated with the Shared Name Utility (sn.exe). The compiler inserts the public key into the assemble manifest and then signs the assemble with the private key.
/keyname: keycontainer or /keyn: keycontainer	Use to create shared components. Keycontainer contains a key pair generated and installed into a key container with the Shared Name Utility (sn.exe). The compiler inserts the public key into the assembly manifest and the signs the assembly with the private key.
/main: entrymethod	Specifies the entry-point method name when converting a module to an executable.
/out: filename	Specifies the output filename.
/type: lib|exe|win or /t: lib|exe|win	Specifies the file format of the output file (lib for library, exe for console executable, and win for win32 executable). The default setting is lib.
/version: major.minor.revision.build	Specifies version information for the assembly. The default value is 0.

©2001 Microsoft Corporation

Assembly Registration Utility

(Gac.exe)

You can use gac.exe to install and uninstall an assembly, as well as to list the content of the GAC. The following table shows some of the common usages of the Assembly Registration Utility:

Option	Description
-L	To list the content of the GAC.
-lc	To list the content of the GAC.
-I filename	To install an assemble with file named filename into the GAC.
-u assmblyname	To uninstall an assembly from the GAC by specifying the assembly name. If multiple versions of the same assembly exist, all of them will be removed unless a version is specified with the assemblyname (i.e. gac –u myAssembly, ver=1.0.0.1).

MSIL Assembler (ilasm.exe)

This tool takes MSIL as input and generates a portable executable (PE) file containing the MSIL and the metadata required to run on the .NET Framework. This is the most useful to vendors who would like to create MSIL-compliant compilers. All they have to do is write the compiler to translate the source language to MSIL. Ilsam.exe will take the second step to put the MSIL content into the PE Format where it can be executed on the .NET Framework. The general syntax for MSIL assembler is:

Ilasm [options] MSILfilename

The following table shows some of the common usages of the assemblers:

Option	Description
/debug	This option ensures that the output PE contains debugging information such as local variables, argument names, and line numbers. This is useful for debug build.
/dll	This option produces a .dll output.
/exe	This option produces an .exe output.
/listing	This option produces a listing of the output on STDOUT.

/output=filename	Filename is the output filename.
/?	This option is used to obtain command-line help.

MSIL Disassembler (ildasm.exe)

This tool extracts the MSIL code from a PE file targeted for .NET Framework. The general syntax for this tool is:

Ildasm [options] PEFilename.

The following table shows some of the common usages of the disasembler:

Option	Description
/linenum	This includes references to original source lines.
/out=filename	The output goes to a file instead of in a GUI dialog box.
/source	This shows original source lines as comments.
/text	The output goes in a console window.
/tokens	This shows metadata tokens as classes and members.

C++ Compiler (cl.exe)

The following table shows some of the common usages of the C++ compiler:

Option	Description
/CLR or /COM	This option flags the compiler to compile .NET runtime managed code.
/entry:methodname	Normally, for C++ managed code, this setting should point to the main entry-point function.
/link	This option combines the compile and link steps.
/out:filename	This option allows for the output filename.

©2001 Microsoft Corporation

C# Compiler (csc.exe)

The following table shows some of the common usages of the C# compiler:

Option	Description
/a.keyfile:keyfilename	Keyfilename specifies the key file that contains the key pair for signing the assembly.
/a.version:version	This option enables versioning of assemblies.
/debug	With this option, the compiler will emit debugging information in the output file.
/define:symbol or /d:symbol	This option is similar to C++. Use this option to define preprocessor symbols.
/doc:docname	Docname is the XML output file for the auto-generated XML comment embedded in C# code.
/help	This option shows the command-line help for the C# compiler.
/main:classname	If there is more than one Main entry in different classes, you will have to specify the Main entry in which class you want the entry point of the application.
/out:filename	This option represents the output filename.
/reference:libname or /r:libname	This option allows single or multiple libraries be included with this compilations. For multiple libraries to be included use a semicolon as the delimiter.
/target:exe|library|winexe or /t:exe|library|winexe	This option allows you to specify the type of the output:exe for console executables, library for DLLs, and winexe for Windows Form Applications.
/unsafe	If you use unsafe keywords in your C# code, you will have to use this option when compiling your source.

Visual Basic Compiler (vbc.exe)

The following table shows some of the common usages of the Visual Basic compiler:

Option	Description
/cls	This option enables common language specification (CLS) checking.
/debug	With this option, the compiler will emit debugging information in the output file.
/define:symbol or /d:symbol	Use this option to define preprocessor symbols.
/help or /?	This option shows the command-line help for the Visual Basic compiler.
/keycontainer:keycontainer	Keycontainer specifies the key container that contains the key pair for signing the assembly. See sn.exe for information on generating the key container.
/keyfile:keyfile	Keyfile specifies the key file that contains the key pair for signing the assembly. Se sn.exe for information on generating the key file.
/main:classname	If there is more than one Main entry in different classes, you will have to specify the Main entry in which class you want the entry point of the application.
/out:filename	This option represents the output filename.
Optionexplicit[+/-]	Turn on or off optionexplicit to enforce explicit or implicit declaration of variables. The default setting is on.
/reference:libname or /r:libname	This option allows single or multiple libraries be included with this compilation. For multiple libraries to be included use a semicolon as the delimiter.
/targe:exe|library\winexe or /t:exe|library|winexe	This option enables versioning of assemblies.

©2001 Microsoft Corporation

PE File Format Viewer (dumpbin.exe)

Dumpbin is not a new utility. However, since .NET Framework stores the IL inside the extended PE Format, this old utility is still very useful for examining the structure of executable or DLLs, as well as listing import and export entries of the binaries. The general syntax for this utility is:

Dumpin [options] PEFilename

The following table shows some of the common usages of the dumpbin usages:

Option	Description
/all	Displays all information from the PE file.
/exports	Displays all the exports from the PE file.
/header	Displays all the header information from the PE file.
/imports	Displays all imports for the PE file.

Type Library Exported (tlbexp.exe)

Type library exporter and importer are the two tools necessary for COM interop. The exporter generates a type library for a .NET Framework assembly so that other COM components can interop with .NET components. The general syntax for tlbexp.exe is:

Tlbexp Assembly Name [options]

The following table shows some of the common usages of the tlbexp.exe:

Option	Description
/nologo	This option suppresses the logo of the tlbexp executable.
/out:filename	Filename is the name of the type library file.
/silent	This option suppresses all messages from the tlbexp executable.
/verbose	This option displays extra information while converting the component.
/? or /help	This option displays the help information for the tool.

Type Library Importer (tlbimp.exe)

As the reverse tool of the type library exporter, the importer generates a .NET proxy component for a COM component so that .NET components can use legacy COM components. The general syntax for tlbimp.exe is:

Tlbimp PEFile [options]

The following table shows some of the common usages of the tlbimp.exe:

Option	Description
/ keycontainer: keycontainer	This option signs the resulting assembly with the private key in the keycontainer. The public key in the keyfile will be used in the assembly manifest. See sn.exe for the keycontainer generation.
/keyfile:keyfile	This option signs the resulting assembly with the private key in the keyfile. The public key in the keyfile will be used in the assembly manifest. Se sn.exe for keyfile generation.
/nologo	This option suppresses the logo of the tlbimp executable.
/out:filename	Filename is the name of the type library file.
/silent	This option suppresses all messages from the etlbimp executable.
/unsafe	This option produces interfaces without .NET Framework security checks.
/verbose	This option displays extra information while converting the component.
/? or /help	This option displays the help information for the tool.

Xml Schema Definition Tool (xsd.exe)

XML Schema Definition (XSD) is useful when working with XML schemas that follow the XSD language. With XSD, you can perform the following transformations.

©2001 Microsoft Corporation

- XDR to XSD
- XML to XSD
- Classes to XSD
- XSD to Classes
- XSD to DataSet

XDR to XSD

To convert an XDR-formatted file to XSD, you would use the following syntax:

xsd [options] file.xdr

Note that the file extension .xdr dictates the conversion from XDR to XSD.

XML to XSD

To convert an XML-formatted file to XSD, you would use the following syntax:

xsd [options] file.xml

Note that the file extension .xml dictates the conversion from XML to XSD.

Classes to XSD

You can convert classes to XSD by specifying the runtime assembly file (.exe or .dll extension) as the filename to the utility. You can also specify a particular type within the assembly you want to convert to XSD using the /type flag. The typename can be a wildcard match. If you omit the /type flag, all types in the assembly will be converted. The syntax follows:

XSD [/TYPE:typename] assemblyFile

Or:

Xsd [/T:typename] assemblyFile

XSD to Classes

To convert XSD back to classes, use the /classes or /c flag. You can specify a particular element in the XSD schema to be converted to a class. You can also specify the language for the class source file. The general syntax follows:

Xsd /CLASSES
/ELEMENT: element
/NAMESPACE: namespace
/LANGUAGE: language /URI: uri file.xsd

or:

xsd /C E:element /N:namespace /L:language /U:uri file.xsd

Note that namespace, language, and uri can be specified only once.

XSD to DataSet

To convert XSD to dataset, use the /dataset or /d flag. Again, you can narrow down to a particular element in the XSD schema to be converted. The general syntax follows:

Xsd /D [/DATASET] file.xsd

Shared Name Utility (sn.exe)

Sn.exe guarantees unique names for shared components because these components will end up in the GAC. Each shared component is signed with a private key and published with the public key. The following table shows some common usages of sn.exe:

Option	Description
/?	This option displays more command-linie help.
-d keycontainer	This option is used to remove the keycontainer from the CSP.
-I keyfile Keycontainer	This option reads the key pair in keyfile and installs it in the key container keycontainer in the Cryptographic Service Provider (CSP).
-k keyfile	This option generates a new key pair and writes it to keyfile.
-v assembly	This option is used to verify the shared name in an assembly.

Web Service Utility (wsdl.exe)

Wsdl.exe helps create ASP.NET Web Services and proxies for their clients. The most common usage of wsdl.exe is to generate proxy classes for web services:

Wsdl /command: proxy
/path:path
/language: language
/namespace: namespace
/out:output
/protocol: protocol

©2001 Microsoft Corporation

The path parameter is a local path to a service-description file or URI where the SDL file can be retrieved. The language parameter specifies the language for the output-proxy source file. It can be C#, VB, or JS. The generated class will be in the specified namespace. The output source file is controlled by the output option. The protocol controls which protocol the proxy will use to communicate with the Web Service. The choices of protocols provided by the .NET Framework are SOAP, HttpGet, and HttpPost. You can also have your own protocol here if you've extended the ProtocolInfoImporter class.

For short name options, use the following.

Wsdl /c:proxy

/pa:path

/l:language

/n:namespace

/o:output

/protocol:protocol

Appendix B:

Common Datatypes

Each of the .NET languages might provide its own keywords for the types it supports. For example, a keyword for an integer in VB is Integer, whereas in C# or C++ it is int; a Boolean is Boolean in VB, but bool in C# or C++. In any case, the integer is mapped to the class Int32, and the Boolean is mapped to the class Boolean in the System namespace. CLS-compliant types are not guaranteed to interoperate with all CLS- compliant languages.

Table c-1. Common data types

Type	Description
Boolean	True or false.
Byte	8-bit unsigned integer: 0 to 255.
Char	Character. Unicode 16-bit character.
DateTime	Represents a date and time value.
Decimal	Can represent positive and negative values with 28 significant digits; 79,228,162,514,264,337,593,543,950 ,335 through - 79,228,162,514,264,337,593,543,950 ,335
Double	Stores 64-bit floating-point values: -1.79769131486321570e308 to 1.79769131486321570e308
Guid	Represents a globally unique identifier (GUID).
Int16	Stores 16-bit signed integers: -32,768 to 32,767
Int32	Stores 32-bit signed integers: -2,147,483,648 to 2,147,483,647
Int64	Stores 64-bit signed integers: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
Sbyte	Represents an 8-bit signed integer. The Sbyte type is not CSL-compliant.

©2001 Microsoft Corporation

	–128 to 127
Single	Represents an IEEE 754f, single precision, 32-bit value: -3.40282346638528859e38 to 3.40282346638528859e38
String	Represents a string of Unicode characters.
Uint16	Represents a 16-bit unsigned integer. The Uint16 type is not CSL compliant. 0 to 65,535
Uint32	Represents a 32-bit unsigned integer. The Uint16 type is not CSL compliant. 0 to 4,294,967,295
Uint64	Represents a 64-bit unsigned integer. The Uint16 type is not CSL compliant. The Uint64 data type can represent positive integers with 18 significant digits: 0 to 184,467,440,737,095,551,615
Void	Void.

Table C-2 shows a number of useful container types that the .NET Framework provides.

Table C-2. Container types.

Type	Description
ArrayList	This class implements the Ilist interface. The array can grow dynamically in size.
BitArray	This class represents a compact array of bit values. Each element represents a Boolean value (true/false).
Hash-Table	This class represents a collection of associated keys and values that are organized based on the hash code of the key.
Queue	This class represents a first-in, first-out collection construct.
SortedList	This class is similar to HashTable except that all elements are sorted

	by their actual keys (not hashed) and elements are accessible through either key or index.
Stack	This class represents a first-in, last-out stack construct.

Usage

This section demonstrates how you can take advantage of container types. We don't illustrate all methods and properties, but we show the important characteristics of these types. All examples in this chapter are in C#; however, you can use these CSL types from any other CLS-compliant languages.

©2001 Microsoft Corporation

Appendix C

Third-Party Languages for .NET

Table A-2 shows a list of third-party languages with compilers that target the CLR. Some of these are research languages, while others are commercial languages that target .NET. Browse the provided web sites to read more about the languages that interest you. As Noted earlier, this list of languages could grow by the time this book hits the market, so be sure to check the following sites for the most up-to-date listings.

Table A-2. Third-party languages

APL - http://www.dyadic.com/

Cobol - http://www.adtools.com/info/whitepaper/net.html

Component Pascal - http://www2.fit.qut.edu.au/CompSci/PLAS//ComponentPascal/

Eiffel - http://dotnet.eiffel.com/

Haskell - http://haskell.cs.yale.edu/ghc/

Mercury - http://www.cs.mu.oz.au/research/mercury/

Oberon - http://www.oberon.ethz.ch/lightning/

Perl - http://aspn.activestate.com/ASPN/NET/index

Python - http://aspn.activestate.com/ASPN/NET/index

Scheme - http://rover.cs.nwu.edu/~scheme/

SmallTalk - http://www.qks.com/

Standard ML - http://research.microsoft.com/Projects/SML.NET/index.htm

TMT Pascal - http://www.tmt.com/net.htm

©2001 Microsoft Corporation

©2001 Microsoft Corporation